コンピュータ理工学特別研究報告書

題目

Raspberry Pi を用いた模型自動車の制御

-LED マーカーを使った運転制御-

学生証番号　344018

氏名　合野　優樹

提出日　平成 29 年 1 月 31 日

指導教員　蚊野　浩

京都産業大学
コンピュータ理工学部

要約

本研究はRaspberry Piを用いて模型自動車の制御を行う.模型自動車の後方を単眼カメラで撮影し,操縦する人が持っているLEDマーカーを検出する.検出された結果から,加速・停止・ステアリング角を決定し軽自動車を操作させる.

本研究で開発したLEDマーカーを検出するアルゴリズムは以下のものである.

1) 入力カラー画像のRed成分を大津の方法で二値化する.
2) 21×21画素のテンプレートの周辺と中央の画素だけを用いて,粗いテンプレートマッチングを行う.
3) 粗く抽出したLED候補の形状が丸くかつ密集していることを確認する.
4) さらに正三角形に配置されていることを確認する.

模型自動車の制御に,LEDマーカーの角度とカメラからLEDマーカーまでの距離を用いる.LEDマーカーが車に近い場合は速度を上げる.LEDマーカーが遠い場合は速度を下げる.このようにすることで,カメラの視界からLEDマーカーを見失わないようにした.実際に走行させた結果,模型自動車はLEDマーカーを見失うことは少なくなり,直線ではスムーズに走行させることができた.しかし,コーナーに入るとどうしてもカメラの視角からLEDマーカーが外れることが多く,操縦者がLEDマーカーをカメラの視角に入るように操作しなければならないという課題がある.

目次

1章 序論

コンピュータは様々な分野で利用されている. ノートパソコンやデスクトップパソコンのようにキーボードやマウスで操作するものだけではなく,洗濯機や冷蔵庫などの家電にも搭載されている.また, 自動車の ECU(Engine Control Unit)やカーナビにも組み込まれている.私たちの身の回りには目に見えない形で数多くのコンピュータがあり, それらは製品に組み込まれている.特定の機能を実現するためにコンピュータが組み込まれた機器を組み込み機器と呼び,その機器を制御するためのシステムを組み込みシステム,そのシステムを動作させるためのコンピュータを組み込みコンピュータと呼ぶ.

自動車には数多くのセンサやコンピュータが使われている. スピードメータやタコメータ,ガソリンの残量計などのセンサは,昔から車に搭載されている.マイクロコンピュータは, 1980 年代にブレーキやエンジンを制御するために利用され始めた. 現在の車は 100 個を超えるコンピュータが搭載されている.最近の技術をみると, メルセデスベンツはレーダセンサとカメラを用いて危険な車線変更を認識するブラインドスポットアシストや,最適な車間距離を自動的に維持するディスタントパイロット・ディストロニック,車線のはみ出しを検出するレーンキーピングアシストなどの機能を実用化している. それだけではなく,自動車と歩行者が衝突した場合にはボンネット後部を持ち上げることで, 歩行者を保護する機能を実用化している.これらには, いずれもコンピュータが使われている. このように自動車を走行させるにはコンピュータが不可欠なものになっている.

自動車メーカーは自動運転技術に力を入れており,多くの会社は, 2020 年前後を目処に製品化すると発表している.自動運転は,人間が無意識に行っている「認知」,「判断」,「操作」の処理を自動化することである.認知とは,歩行者や車両を的確に検出し,周囲の状況を理解することである.例えば,カメラなどの環境センサを用いて物体検出を行ったり,GPS(Global Positioning System)などの位置センサを用いて自車の位置を認識したりすることである.ミリ波レーダやレーザを用いたセンサは周囲の車両などとの距離を測定するために利用される.ミリ波レーダやレーザで障害物の発見はできるが,障害物の種類を特定することは難しい.それに対して, カメラは周辺の情報をより正確に読み取ることができ,障害物の種類を特定することが可能である.

カメラを用いた運転支援技術としてよく知られているものに,スバルのアイサイトがある. これは 2 つのカメラからなるステレオカメラで,三角測量の原理に基づいて対象物までの距離を測定する技術である.ステレオカメラは 2 台のカメラを使うため, 単眼カメラのシステムに比べて, コストが高くなる.

モービルアイ社は単眼カメラを使った運転支援技術を実用化している．これは,車のフロントガラスに取り付けた単眼カメラから画像情報を読み取り,独自のアルゴリズムが前方を走る車や人,車線を認識する．また，対象物との距離を算出し，対象物が危険であれば運転手に知らせる．場合によっては自動的にブレーキを作動させる.モービルアイ社の技術を見ると，単眼の車載カメラであっても，周囲環境に関する豊富な情報を推定可能であることがわかる.

単眼車載カメラを使った自動運転は，前方車両や信号・車線・標識などを認識しながら，車の走行を自動制御することである．そこで利用される画像認識技術は，非常に高度なものである．今回の研究では，その基礎的な検討として，平板に３個の LED を配置した LED マーカーをカメラに観察させ,その状態を認識させることにした．そして,LED マーカーの状態に応じて，車の走行を制御する.

今回の研究で実際の自動車を使うことは難しい.そのため，カメラモジュールを接続した Raspberry Pi を車載カメラと車載コンピュータに見立て，それを使って，TAMIYA 製の電動模型自動車を制御した．そして，LED マーカーを車載カメラに観察させることで模型自動車を遠隔操作する研究を行った．LED マーカーは黒いボードに赤い LED ランプ 3 個を正三角形になるように点灯させたものを使った.この LED マーカーを左右に傾けたり,カメラに近づけたり,カメラから遠ざけるという操作を行うことで,模型自動車を制御して走行させるプログラムを開発した.

2 章　研究に用いた模型自動車システム

本研究用では，自作の組み込みシステムで市販の模型自動車を制御する実験を行った．組み込みコンピュータとして PIC マイコンと Raspberry Pi を利用した．模型自動車はタミヤ RC カーのシャーシ MF-01X（図 1）を利用した．これを通常の RC カーとして利用する場合には，電波受信機から前輪用サーボモータ（前輪の方向角度を変える）と ESC（Electric Speed Controller）に PWM 信号を送り，車体の動きを制御する．今回は，前輪用サーボモータと，前進・後退速度を制御する ESC に送る 2 組の信号を PIC マイコンから制御する．そして，Raspberry Pi から PIC マイコンに I2C インタフェースを介して指令を送ることで，模型自動車を動作させる．

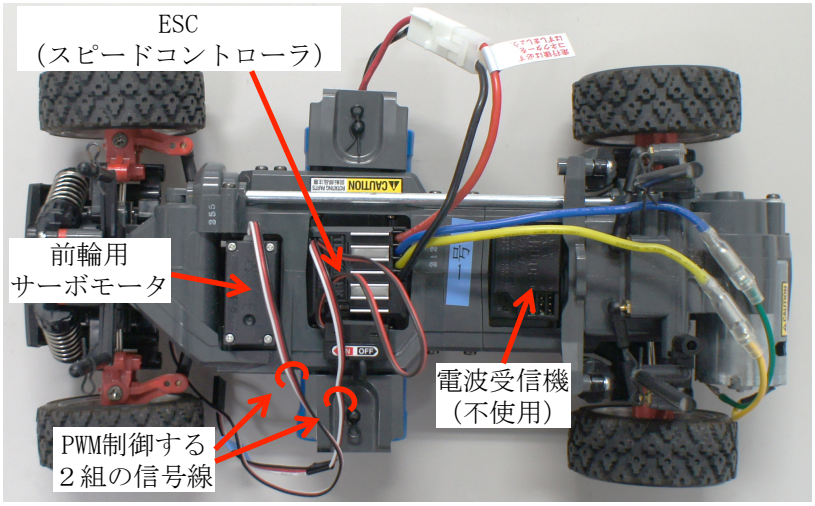


図 1　タミヤ RC カーのシャーシ MF-01X

サーボモータと ESC に加える信号は，いずれも，図 2 のような PWM 制御のための周期的矩形波である．ここで，パルス幅は概ね 1.5msec を中心にして 1.1msec～1.9msec の範囲で制御するようになっている．ステアリング（前輪の方向角度）を制御する場合，1.5msec のパルス幅で前輪の方向角度が中央になる．1.1msec あるいは 1.9msec のパルス幅で左右の最大角度になる．前進・後退の駆動制御に使う場合，1.5msec のパルス幅でニュートラル状態になる．1.5msec よりも短くすると前進，1.1msec で最大の回転数で前進，1.5msec よりも長くすると後退になる．なお，パルス電圧の大きさは 6.0V 程度である．

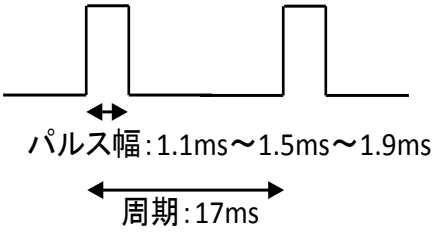


図 2　サーボモータと ESC に加える PWM 信号

図 2 の PWM 信号を生成するために PIC マイコンを利用した．図 3 に PIC マイコンと RC カーの接続を示す．PIC マイコンには PIC16F819 を利用した．PIC マイコンのプログラムを付録 2 に示す．また，図 3 に示すように，PIC マイコンは I2C インタフェースを介して Raspberry Pi とも接続されており，PWM 信号を適切な状態に設定するためのコマンドを入力することができる．このように構成することで，Raspberry Pi で開発したプログラムから模型自動車を制御することが可能になった．

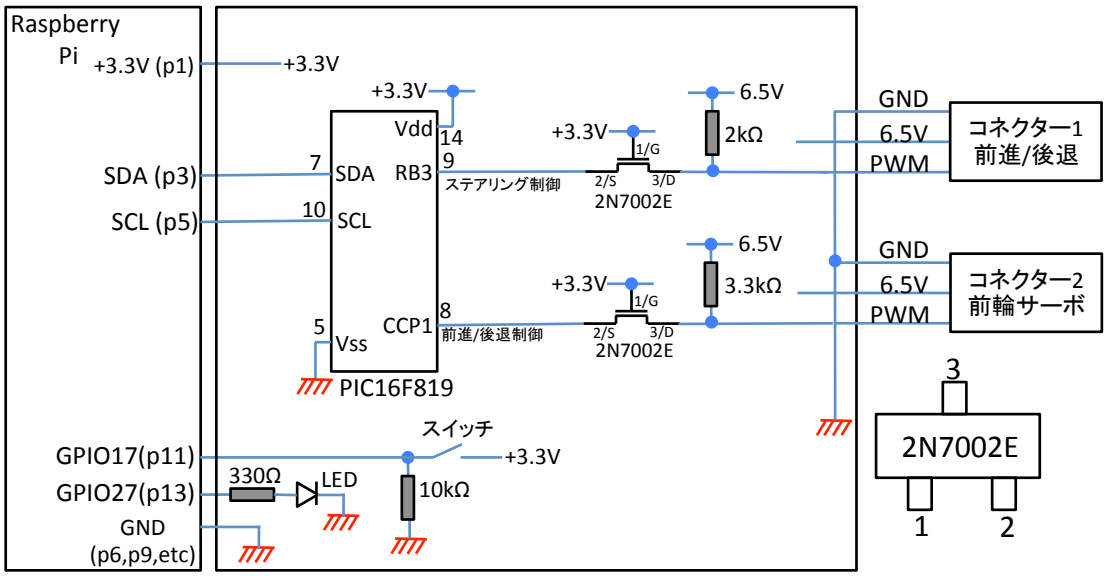


図 3　Raspberry Pi と RC カーの ESC とのインタフェース回路

Raspberry Pi は，名刺サイズのコンピュータで，Linux 系の OS が動作する．ディスプレイやネットワークに接続することで，通常のパソコンとして利用可能である．また，汎用入力端子(GPIO)を用いて回路や装置を接続することが容易で，組み込み機器を制御することができる．今回は一組の GPIO 端子を I2C インタフェースの信号線として利用して，PIC マイコンと接続した．

Raspberry Pi 用に専用のカメラモジュールが販売されている．5 メガピクセルの画像センサとレンズから構成されており，2592×1944 画素の静止画と，1920×1080 の動

画を撮影することができる．今回の研究では，正三角形に配置した LED マーカーの画像をカメラで撮影し，LED の位置と動きによって模型自動車を制御した．図 4 に模型自動車システムの外観と動作状態を示す．Raspberry Pi のプログラムは C/C++で開発し，画像処理ライブラリとして OpenCV を利用した．

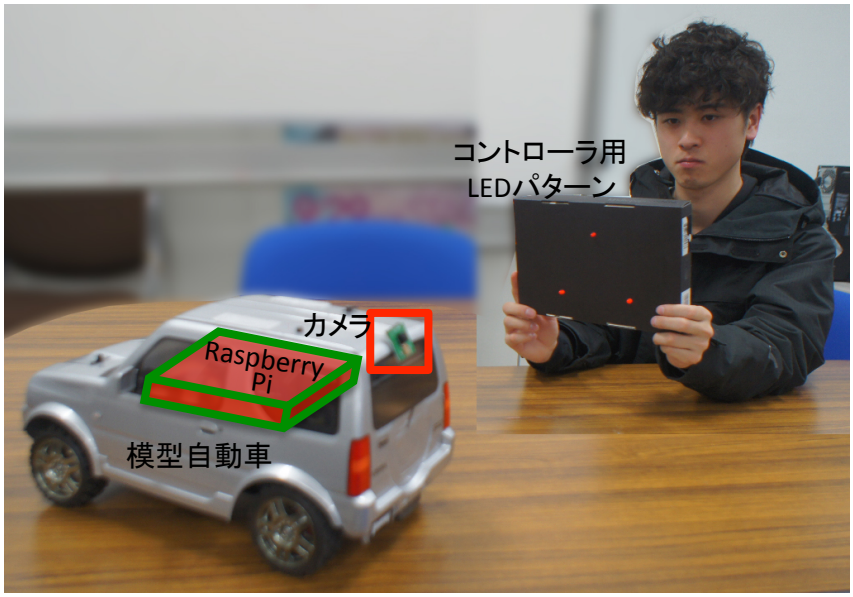


図 4　模型自動車システムと LED マーカーの外観

3 章　LED マーカーの検出

3.1　検出手法の概要

自動車の操作に使う LED マーカーを，模型自動車のカメラから撮影した画像を図 5 左に示す．本研究では，３箇所の LED の位置情報を利用して模型自動車を制御する．そのためには，これらの位置を高速かつ頑健に検出する必要がある．

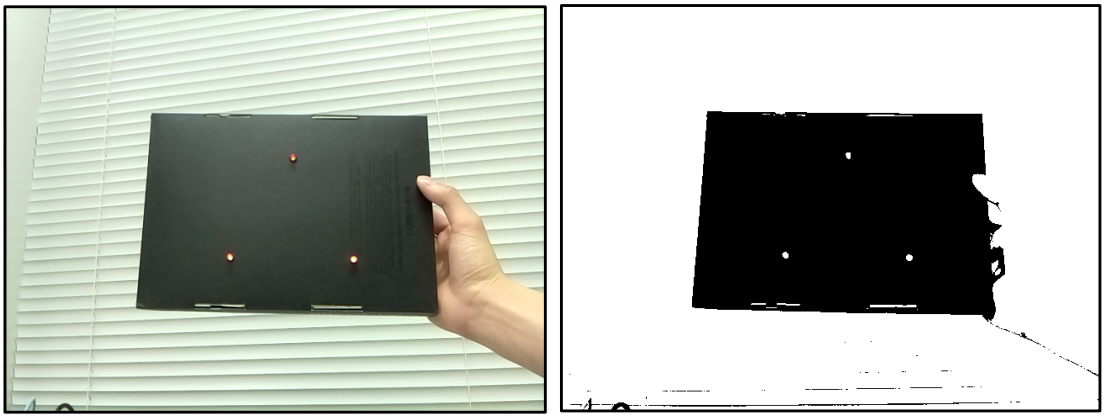


図 5　カメラに取り込まれた LED マーカーの元画像と２値画像

図 5 を用いて，今回開発した方法の概要を説明する．LED マーカーの画像は 640 画素×480 画素のカラー画像として入力する（図 5 左）．これを図 5 右のように２値化した後，LED の位置を検出する．LED マーカーを検出するために RGB の各成分に分解する．本研究では赤い LED を検出するため，Red 成分画像を大津の法則を用いて二値化している．

初期状態では LED の位置は不明であるから，画像全体から LED の候補位置をいくつか検出する．その具体的な方法は 3.2 で説明する．検出した LED の候補位置の中に，正三角形に近い３箇所が有れば検出成功とする．検出候補が３箇所未満の場合と，正三角形に近い３箇所がない場合は検出失敗とする．３箇所の LED マーカーの検出に成功した場合，次の制御ループでは，検出した位置の近傍で LED の候補位置を検出する．検出に失敗した場合，再度，画像全体から候補位置を検出する．以降の処理は同様である．このように LED マーカーの検出に成功した場合には，連続する制御ループにおいて，その近傍のみを探索することで，高速に追尾することを可能とした．

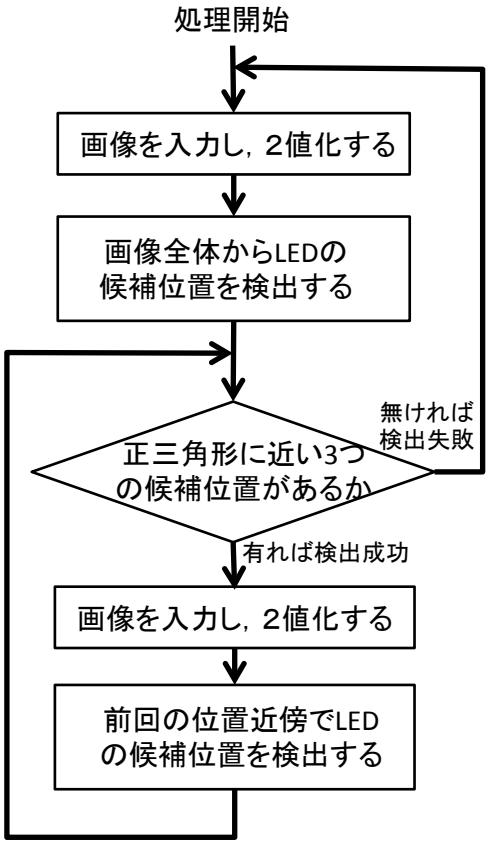


図 6 LED マーカーの位置を検出する処理の流れ

3.2 LED の候補位置の検出

LED の候補位置を抽出するアルゴリズムをいくつか検討した．最も単純なテンプレートマッチング法は計算コストが大きく，しかも，LED の見え方が変化した場合に検出に失敗する可能性が高かった．

開発した方法を，図 6 を用いて説明する．この方法は，21×21 画素の小領域が LED の候補領域であることを二段階で判定する．第一段階は，図 7 左において灰色で示す四辺の 80 画素が黒画素で，かつ中央の 4 画素が白画素であることを確認する．この判定は，21×21 画素の 19%だけを使うため，通常のテンプレートマッチングに比べて高速に実行することができる．これに合格した場合，第二段階の判定を行う．

第二段階では，領域内の白画素集合に内接する長方形の長辺（long），短辺（short），白画素数（count）を求め，(short/long)× count/(short・long)によって LED らしさを判定した．ここで(short/long)の項で内接長方形の正方形らしさを測り，count/(short・long)の項で白画素集合の密集度を測っている．白画素集合が正方形で隙間なく密集している場合，LED らしさは 1 になり，円形で密集している場合，0.8 程

度になる．今回の研究では，この数値が 0.3 以上の場合に LED 候補の検出に成功したと判定し，その値を LED らしさの度合いとした．また白画素集合の重心を LED 候補位置とした．

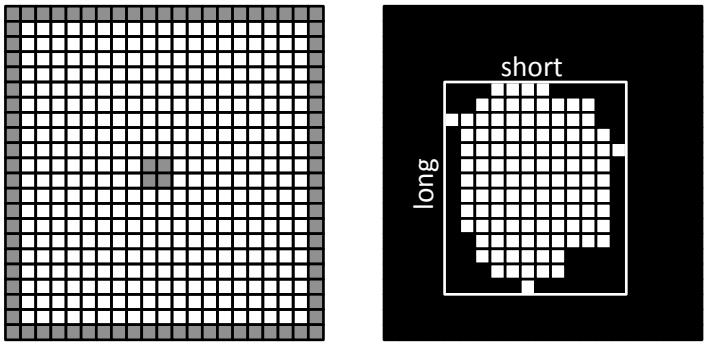


図 7 21×21 画素の領域を LED の候補領域と判定するための説明図

3.3 LED 候補位置の検証

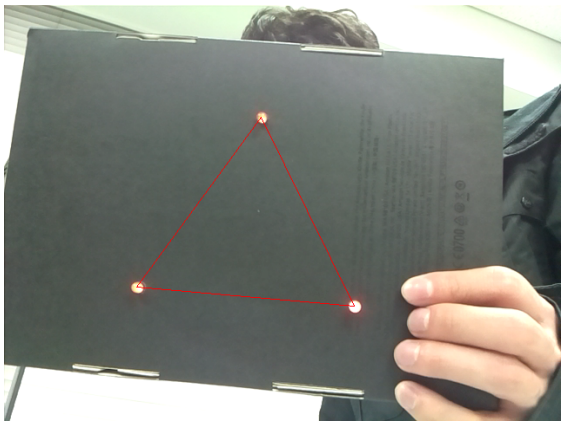
3.2 の手順で検出した LED の候補位置の中に，正三角形に近い３箇所が有れば LED の検出に成功したとする．その方法について説明する．

まず,候補位置が 3 個未満であれば,検出失敗とする.候補位置が３個以上ある場合，LED らしさの度合いが最も高い３点を求める． ３点がつくる３辺の長さの２乗を求め，その最小と最大の比が 0.7 以上であれば正三角形に近いと判断し，３箇所の LED が検出できたと判定した.

3.4 検出手法の性能

3.1～3.2 の方法で LED マーカーを検出した場合の LED マーカーの検出性能について説明する.

今回作成したプログラムは車載カメラと LED マーカーの距離が約 20cm～135cm 間で正しく検出できた.図 8 に 20cm と 135cm に LED マーカーをおいた場合の画像を示す.20cm の距離では LED マーカーの点灯部分が直径 15～20 画素程度になった． 135cm の距離では点灯している部分が直径 2～4 画素程度であった.

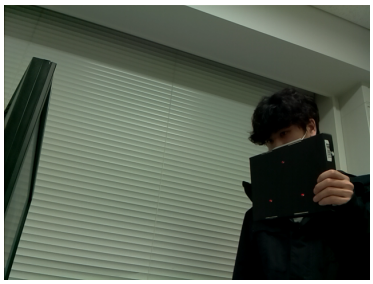



(a) 20cm での LED マーカー　　(b) 135cm での LED マーカー

図 8 20cm から 135cm の距離で観察した LED マーカーの画像

次に，車載カメラから見た LED マーカーの角度と検出能力を確認した．図 9 は距離 80cm に LED マーカーを置き，垂直にした時と，40°，45°の場合について検出結果を示す．垂直と 40°の場合には 3 つの LED が 3.3 で述べた判定条件に合格した．45°の場合には，判定に合格しなかった．このことから，LED マーカーを垂直状態から 40°以内で利用する必要がある．





(a) 垂直な LED マーカー　　(b) 40°の LED マーカー　　(c) 45°の LED

図 9 80cm に置いた LED マーカーを垂直，40°，45°で観察した結果

LED マーカーの検出速度を検証した．システムの動作確認を行うために，カメラから読み込まれた画像をディスプレイに表示しなければならない．ディスプレイに画像を表示している場合の LED マーカーを画像全体から探索している時の処理速度は約 5〜6fps であり，連続的に LED マーカーの追尾に成功している場合だと約 10〜11fps となった．このような結果になった理由は，3.2 で説明したように，画像全体からテンプレートマッチングを行うと計算コストが大きく処理に時間がかかる為である．また，連続的に LED マーカーの追尾に成功している場合はその周辺のみをテンプレートマッチングさせている為，画像全体から探索している時と比べると大きく処理速度が早くなっている．

しかし,10〜11fpsではスムーズに模型自動車を操作するには処理速度が遅い.その為,車載カメラから読み込まれた画像をディスプレイに表示せず操作するプログラムを作成した. 画像をディスプレイに表示せず操作するプログラムでは,LEDマーカーを画像全体から探索している時の処理速度は約7〜8fpsであり,連続的にLEDマーカーの追尾に成功している場合だと約20〜24fpsとなった.ディスプレイに車載カメラから取り込まれた画像を表示させないことによって連続的にLEDマーカーの追尾に成功している場合の処理速度が2倍程度になり,操作がよりスムーズにできるようになった.

表1 LEDマーカーの検出速度

表示	探索	速度(fps)
あり	全体	5〜6
あり	近傍	10〜11
なし	全体	7〜8
なし	近傍	20〜24

4 章　LED マーカーを使った模型自動車の制御

4.1　LED マーカーを使った操作

通常の自動車はハンドル（ステアリング）とアクセル・ブレーキ（加減速）を使って操作する．これと同様の操作を LED マーカーで実現した．

加減速を次のように制御した．

1） 検出した LED マーカーを三角形で表現する(図 10)．

2） カメラから見た底辺の長さを測定する．

3） 底辺の長さからカメラと LED マーカーの距離を推定する．

4） 推定した距離が閾値よりも近ければ加速し，遠ければ減速する．

5） LED マーカーが検出できなかった場合，停止する．

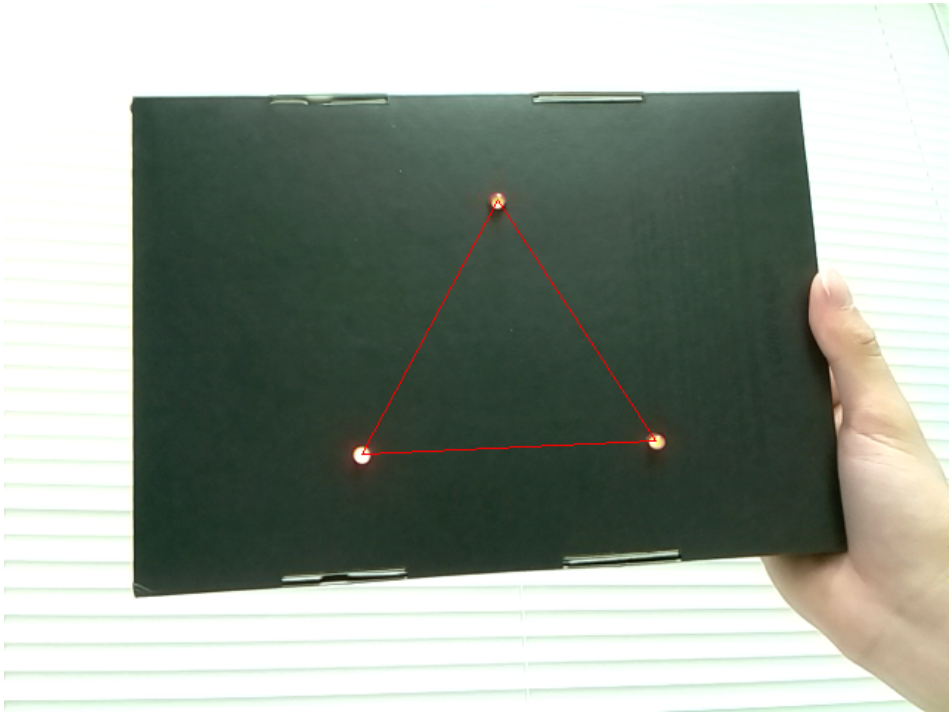


図 10　三角形で表現した LED マーカー

ステアリングを制御するために，最初に考えた方法を説明する．

1） 検出した LED マーカーを三角形で表現する(図 10)．

2） 三角形の底辺と画像の横軸の角度を測定する．

3） 測定した角度からステアリング角を決定する．

この方法の問題は，LED マーカーが 60° よりも大きく傾いた場合，三角形の底辺に対応する辺が入れ替わってしまうことである．その結果，ステアリング角のもとになる底辺角度の数値が急激に変化し，ステアリング操作が不安的になった．

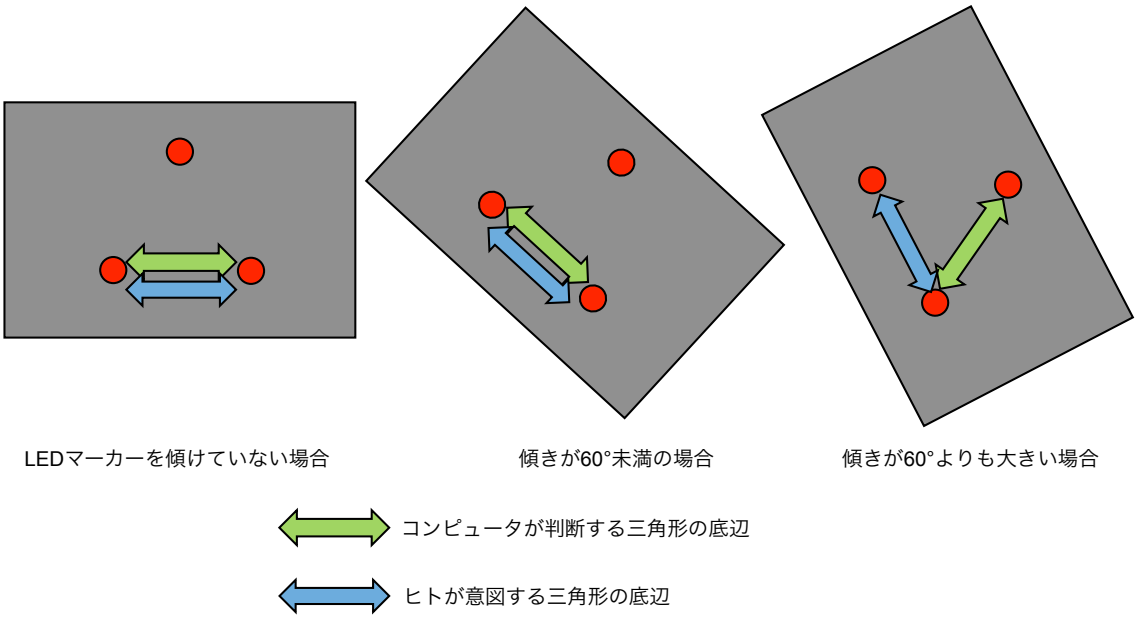


図１１　LED マーカーを 60°よりも大きく傾けた場合の底辺の入れ替わり

これを改善するには次のような方法がある.

① 底辺角度とステアリング角の対応を，底辺角度が 60°に近づく前にステアリング角が最大になるように設定することで，LED マーカーの回転操作を狭い範囲にする.

本実験では,LED マーカーの三角形の底辺角度が 40°の時にステアリング角を最大になるように設定した.LED マーカーの回転操作を狭い範囲にすることによって,LED マーカーを 60°以上傾けても最初の方法のようにステアリング角が急激に変化することがなくなった.この方法では 40°を限界としているので 80°まで LED マーカーを回すことが可能であるが,それ以上回すと最初の方法と同じ原理でステアリング操作が不安定になるという問題がある.

② 3 個の LED マーカーの x 座標値の比率からステアリング角を求める.

図 12 に LED マーカーが少し傾いた状態を示す.3 個の LED マーカーの x 座標を x_1, x_2, x_3 とする．これらの座標値を $x_1 <= x_2 <= x_3$ となるように並べ直す．$l = x_2 - x_1$，$m = x_3 - x_2$ を求め，l と m の比率からステアリング角を求める．この方法では，２つの LED が横に並んだときにステアリング角がほぼニュートラルになり，2 つの LED が縦に並んだ時に，ステアリング角が最大になる．この方法には，LED マーカーをステアリング角最大の位置を超えて回転させると，ステアリング角が徐々の小さくなるという問題はある.

本研究では①の方法だと②の方法に比べ LED マーカーを大きく回しても操縦者が意図しない動きになりにくいと考えたため,①の方法でステアリング角を決定した.

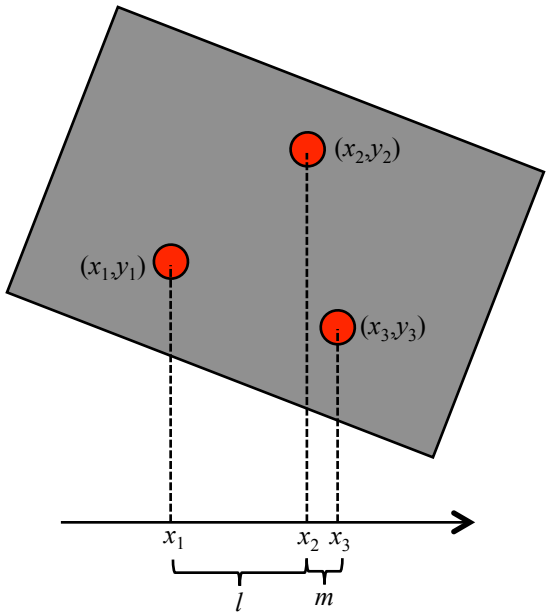


図 12 LED マーカーの傾きと x 座標の値

4.2 ステアリング・加速の制御の実験

3 章の方法を用いて LED マーカーを観察させ, 模型自動車を制御することによって 14 号間 2 階を壁や障害物と接触することなく一周させることができた. 模型自動車の速度と LED マーカーまでの距離の関係は表 2 のようになっている. 模型自動車は前後進 30 段階の制御が可能であるが, 最高速で走行させるのは危険である為, LED マーカーで操作できる最高速は 30 段階中 10 段階目となっている. また, 模型自動車のタイヤ角は調節が可能である左右 15 段階, 全てを使用している.

表 2　模型自動車の速度と LED マーカーの速度の関係

カメラから LED マーカーの距離	模型自動車の速度
0cm～25cm	10
25cm～50cm	9
50cm～75cm	8
75cm～100cm	7
100cm～125cm	6
画面の上端に LED マーカーがある場合	10
画面の下端に LED マーカーがある場合	0

表 3　模型自動車のタイヤ角と LED マーカーの傾き角度の関係

LEDマーカーの傾き角度	模型自動車のタイヤ角	LEDマーカーの傾き角度	模型自動車のタイヤ角
40° ～ 37.5°	15	-40° ～ -37.5°	-15
37.5° ～ 35°	14	-37.5° ～ -35°	-14
35° ～ 32.5°	13	-35° ～ -32.5°	-13
32.5° ～ 30°	12	-32.5° ～ -30°	-12
30° ～ 27.5°	11	-30° ～ -27.5°	-11
27.5° ～ 25°	10	-27.5° ～ -25°	-10
25° ～ 22.5°	9	-25° ～ -22.5°	-9
22.5° ～ 20°	8	-22.5° ～ -20°	-8
20° ～ 17.5°	7	-20° ～ -17.5°	-7
17.5° ～ 15°	6	-17.5° ～ -15°	-6
15° ～ 12.5°	5	-15° ～ -12.5°	-5
12.5° ～ 10°	4	-12.5° ～ -10°	-4
10° ～ 7.5°	3	-10° ～ -7.5°	-3
7.5° ～ 5°	2	-7.5° ～ -5°	-2
5° ～ 2.5	1	-5° ～ -2.5	-1
-2.5° ～ 2.5°	0	-40° ～ 40°	直前のタイヤ角

4.3 考察

自動車に求められている画像処理技術は完全に近い精度で誤認識を発生させてはならない.

電動模型自動車に搭載されたカメラで LED マーカーを観察し,LED マーカーを用いて操作させた.LED マーカーを正確に読み取ることができれば,操縦者が意図している方向に走行させるのである.しかし,カメラの視角が限られているために右左折時に LED マーカーが画角から外れることも多く,角度のついたコーナーをスムーズ曲がることができなかった.

5章　結論

カメラモジュールに接続した Raspberry Pi を TAMIYA 社の電動模型自動車に搭載し,LED マーカーを搭載されたマーカーで観察することによって,LED マーカーを用いて電動模型自動車を操作することができた.また,操縦者と模型自動車の速度が異なる際に近づきすぎたり遠すぎたりすること,模型自動車を急旋回させることによってカメラの画角から LED マーカーが外れてしまった.また,一般的なテンプレートマッチングとは異なり,中心の４画素と周辺の 80 画素以外はどのような値でも候補としてあげることによって,カメラから取り込まれた画像が歪んでいても検出させることができ,検出率を上げることができた.

しかし,自動車に用いられる画像処理技術の精度は非常に高いものでなくてはならない.また,本実験では室内で行なったが,実際の自動車では室外はもちろん,霧・雨・逆光といった様々なケースで走行しなくてはならない.本研究を通して命を預かる自動車のコンピュータ技術は絶対的な信頼ができるシステムでなければならないと感じた.

参考文献

[1]名刺サイズの魔法のパソコン ラズベリー・パイで遊ぼう！　林 和孝
株式会社ラトルズ pp.134~147, 2016 年

[2] http://opencv.jp/reference_manual

謝辞

本論文を作成にあたり,丁寧な御指導を賜りました蚊野浩教授に感謝いたします.

付録１　Raspberry Pi で開発したプログラムの説明

	機能
Rasled.cpp	模型自動車に搭載された車載カメラの入力画像の中から LED マーカーを検出し,その LED マーカーの操作に対して模型自動車を制御するプログラム

付録 2　PIC マイコンのプログラム

```
#include <16f819.h>

// CONFIG

#fuses NOPROTECT,CCPB2,NODEBUG,NOWRT,NOCPD,NOLVP,NOBROWNOUT,PUT,NOMCLR,NOWDT,INTRC
// CONFIG, DEBUG（NODEBUG ではなく）, MCLR（NOMCLR ではなく）
//#fuses NOPROTECT,CCPB2,DEBUG,NOWRT,NOCPD,NOLVP,NOBROWNOUT,PUT,MCLR,NOWDT,INTRC

//内部クロック 1MHz で利用する。
#use delay(internal = 1MHZ) //(8MHZ,4MHZ,2MHz,1MHZ,500kHz,250kHZ,125kHz,31kHz)

// I2C の定義。Raspberry Pi 側からみたアドレスは 0x2f（0x5e を 1bit 右シフトした値)。
#use i2c(SLAVE, SDA=PIN_B1, SCL=PIN_B4, ADDRESS=0x5e, FAST, FORCE_HW)
#use fast_io(B) //固定入出力モード

//PWM のパルス幅に関する定数、
#define Width1_ctr 367 //タイマー1（クロック 4usec）は 365 をセットして、1.5msec 程度
#define Width2_ctr 85 //タイマー2 は 85（クロック 16usec）をセットして 1.5msec 程度
#define Width1_max  (Width1_ctr + 30) //±50 で±0.2msec
#define Width1_min  (Width1_ctr - 30)
#define Width2_max  (Width2_ctr + 15) //±12 で±0.192msec
#define Width2_min  (Width2_ctr - 15)

#define ACK 1

//タイマー1、タイマー2 に設定する値用のメモリ
int16 Width1 = Width1_ctr;
int8 Width2 = Width2_ctr;

/*
タイマー0 で 17msec ごとに割込みを発生させ、
 * 割込み発生後、タイマー1 とタイマー2 に設定した時間幅のパルスを生成する
 */
#INT_TIMER0
void intTimer0() {
    //タイマー0 による割込みが発生していることを確認するために 18 番ピンをトグルする
    //output_toggle(PIN_A1);

    //次の割込みのためにカウンタをセットする。
    set_timer0(122);

    //CCP1 に 4us×Width1 のパルスを発生させる
    setup_ccp1(CCP_OFF);    //CCP1（8 番ピン）を Low にする
    setup_ccp1(CCP_COMPARE_CLR_ON_MATCH);   //カウント値が WIdth1 にマッチすれば CCP1
を Low にする
    CCP_1 = Width1;
    set_timer1(0);
    //タイマー2 のクロックは 4us×4=16us
    setup_timer_2(T2_DIV_BY_4, Width2, 1);
    set_timer2(0);
    enable_interrupts(INT_TIMER2);  //タイマー2 の割込みを許可する。
    output_high(PIN_B3);            //9 番ピンを High にする。
}

/*
 * タイマー2 による割込みルーチン。9 番ピンを Low にする。
 */
#INT_TIMER2
void intTimer2() {
    output_low(PIN_B3);
```

```
61          setup_timer_2(T2_DISABLED, 0, 1);
62      }
63
64      /**** main 関数 ****/
65      void main() {
66          setup_oscillator(OSC_1MHZ);
67          //output_low(PIN_A1); //動作確認のために 18 ピンを High にする。
68
69          //ポート B の全端子の入出力を設定する。
70          set_tris_b(0xf3);
71
72          //17msec 周期で発生する割込み用にタイマー0 を使用する
73          //タイマー0 のクロックは 4us×32=128us、カウンタは 8bit
74          setup_timer_0(RTCC_INTERNAL | RTCC_DIV_32 | RTCC_8_BIT);
75          //17ms/128=133 である。タイマーには 255-133=122 をセットする・
76          set_timer0(122);
77
78          //タイマー1 のクロックは 4us
79          setup_timer_1(T1_INTERNAL | T1_DIV_BY_1);
80
81          //タイマー0 による割込みを許可する
82          enable_interrupts(INT_TIMER0);
83          enable_interrupts(GLOBAL);
84
85          /* Raspberry Pi 側は wiringPi の I2C 関数 wiringPiI2CWrite(fd,data)を使ってデータを
86           * 送ってくることを想定している。この場合、1 回の wiringPiI2CWrite()の呼び出しで
87           * まず、デバイスアドレスの 1 バイト、続いて data の 1 バイトが送られる。下記のプロ
88      グラム
89           * ではデバイスアドレスは読み飛ばしている。
90           */
91          signed int8 rcvData;
92          while(1) {
93              if (i2c_poll()) {
94                  rcvData = i2c_read(ACK);
95
96                  if (rcvData == 0x5e) {
97                      //NOP
98                  }
99                  //下位 2 ビットにコマンドの種類、上位 6 ビットに数値がセットされている
100                 else if ((rcvData&0x03) == 0x01) {
101                     Width1 = Width1_ctr + ((rcvData&0xfc)/4);
102                     if (Width1 > Width1_max) Width1 = Width1_max;
103                     else if (Width1 < Width1_min) Width1 = Width1_min;
104                 }
105                 else if ((rcvData&0x03) == 0x02) {
106                     Width2 = Width2_ctr + ((rcvData&0xfc)/4);
107                     if (Width2 > Width2_max) Width2 = Width2_max;
108                     else if (Width2 < Width2_min) Width2 = Width2_min;
109                 }
110             }
111         }
112     }
```